Chapter 19: Using template_product to Build and Distribute UPS Products

In this chapter we describe the **template_product** product, and show how to use it to build and distribute a product.

19.1 Overview

To simplify and somewhat automate the process of building **UPS** products, we have designed the product **template_product**. Once this product is installed on your system, it can be cloned into a new product area and "turned into" the new product. **template_product** can be used to build products of all types (shell script, pre-built binary, source code).

The following is a summary of the steps involved when using **template_product** to build a **UPS**-compatible product. Each step is described in detail later in this chapter:

1) Make sure **template_product** is installed on your system; install it if necessary

2) Setup `template_product`

3) Create a directory for your product

4) Clone **template_product** to create a template for your product in the new directory

5) Insert the product into the template

6) Setup and test the product

7) Distribute the product (using the Makefile provided with **template_product**)

Also discussed in this chapter are:

- customizing a tar file
- adding a product to a **CVS** repository
- removing a product from a distribution node using the provided Makefile

19.2 Accessing template_product

The **template_product** product may already be installed on your system. If not, download it from the distribution node and install it into the main products area on your system by using the usual installation commands:

```
% setup upd
% upd install template_product
```

19.3 Cloning template_product

Next you need to setup **template_product**, make a directory to hold your new product, and clone **template_product** into this new area using a script that comes with it called `CloneTemplate`. You need to provide the name and version of your product to this script (we use **newprod** v1_0 in this example). Enter this sequence of commands:

```
% setup template_product
% mkdir /tmp/newprod
% cd /tmp/newprod
% CloneTemplate
   Product name? newprod
   Product version? 1.0
   Platform specific product [yN]? y
   Dependant products [list as fred:joe:harry]?
   installing template product files in /tmp/newprod
   /newprod
   /tmp/newprod/.
   /tmp/newprod/.header
   /tmp/newprod/.manifest.template_product
   /tmp/newprod/ups
   /tmp/newprod/ups/Version
   /tmp/newprod/ups/INSTALL_NOTE.template
   /tmp/newprod/ups/template_product.table
   /tmp/newprod/ups/.manifest.template_product
   /tmp/newprod/Makefile
   /tmp/newprod/test
   /tmp/newprod/test/TestScript
   /tmp/newprod/README.template
   42 blocks
   Customizing product as newprod...
```

```
16955
# for Flavored products
?
 for NULL products
 for NULL products
#   QUALS is added qualifiers, like: "QUALS=mips3:debug"
#
      UPS_SUBDIR=ups
```

```
# for Flavored products
         FLAVOR=$(DEFAULT_FLAVOR)
          QUALS=""
# for NULL products
#         FLAVOR=$(DEFAULT_NULL_FLAVOR)
#          QUALS=""
##-------------------------------------------------------------
--------
## Files to include in Distribution
16957
```

The files listed in the command output have now been copied into the new product directory, and `Makefile` and `ups/template_product.table` have been customized/renamed for the product. Note that the output shows the full pathname to the created files even though you are working from within this new product directory.

19.4 The Top-Level Makefile

The cloning of **template_product** creates a Makefile in the new product's root directory, e.g., `/tmp/newprod/Makefile`. In order for this Makefile to know what it needs to about the new product, you generally need to make a few changes to the top page or so, e.g., change the flavor, add build instructions, and so on. Changes of this type are discussed in section 19.6.3 *Add Build Instructions to Top-Level Makefile*. You can also add commands to other targets.

The first part of the file is reproduced here for reference (comments not shown):

```
             SHELL=/bin/sh
               DIR=$(DEFAULT_DIR)
              PROD=newprod
      PRODUCT_DIR=MYPROD_DIR
              VERS=v1_0
   TABLE_FILE_DIR=ups
       TABLE_FILE=newprod.table
            CHAIN=development
       UPS_SUBDIR=ups
  ADDPRODUCT_HOST=fnkits.fnal.gov
DISTRIBUTIONFILE=$(DEFAULT_DISTRIBFILE)
           FLAVOR=$(DEFAULT_FLAVOR)
               OS=GENERIC_UNIX
            QUALS=
             CUST=none
...
```

```
#-----------------------------------------------------------
-------
all: proddir_is_set build_prefix

clean:
        rm -f $(PREFIX)

spotless:

test: proddir_is_set clean FORCE
        sh test/TestScript
...
```

19.5 Inserting your Product into the Template

Now you need to add your actual program into the **template_product** clone, and run build instructions, if any. For shell scripts and pre-built binaries, all you need to do is create a `bin` directory under the product root, and put the executable in it. For source code, you need to first create a `src` directory under the product root, put the source file in it, and then build the product as described in the next section, 19.6 *Building the Product*.

19.6 Building the Product

19.6.1 Add Build Instructions

We recommend that you create a Makefile (separate from the one provided) to ensure reproducibility of the build procedure. Create or copy the Makefile in the `src` directory, and include a build target, e.g., `install`, as shown (again, we use **`echo`** to create the file since it's very simple for this example):

```
% echo "install:; cp hello ../bin" > Makefile
```

19.6.2 Run the Initial Build

Now create the `bin` directory under the product root, and run **`make`** to complete the build:

```
% mkdir ../bin
% make hello install
```

```
cc    -o hello hello.c
cp hello ../bin
```

19.6.3 Add Build Instructions to Top-Level Makefile

Now it's time to customize the top-level Makefile created by `CloneTemplate` (refer to section 19.4 *The Top-Level Makefile* for a partial file listing). Typical macro definitions that need to be changed for a compiled program are:

```
FLAVOR=$(DEFAULT_FLAVOR)
    OS=$(DEFAULT_OS)
 QUALS=
  CUST=$(DEFAULT_CUST)
```

Next, add the build instructions under the `all` target. For this example, they are the two commands that were just run (**`mkdir`** and **`make`**).

```
all: proddir_is_set build_prefix
        -mkdir bin
        cd src; make hello install
```

19.6.4 Rebuild Instructions

The next time this product requires a build, you would just run the command:

```
% make [all]
```

from the product root directory.

19.7 Testing your Product

Now you can setup and test your product. As an example, for our product we might run:

```
% setup newprod v1_0 -r $cwd -M ups -m newprod.table
```

or, for Bourne shell,

```
$ setup newprod v1_0 -r ‘pwd‘ -M ups -m newprod.table
```

followed by:

```
% hello
   hello world
% unsetup newprod v1_0
% hello
```

```
sh: hello: command not found
```

After testing, edit the `test/TestScript` file so that it tests your software. In many cases, writing a good test script can be rather challenging. Include at least a basic test to ensure that the product works properly. For our example, the test script just needs to run our **hello** program and verify its output, e.g.,:

```
#!/bin/sh
hello | grep "hello world" > /dev/null
```

This will exit with a successful exit code if **hello** prints `hello world`, and fail otherwise.

19.8 Customizing your Tar File

Products generally get distributed as tar files. The **template_product** top-level Makefile can be used to make a product tar file and add it to the distribution node in one step. There are several variables in the Makefile that control what **template_product** includes in the tar file it makes of a product:

`ADDDIRS="<dir1> <dir2> <dir3>..."`

lists directories whose non-**CVS**-bookkeeping-files should be added. The default is for this to be set to "`.`", the current directory, and the other variables left blank. If you only wanted to include the `bin` and `lib` directories of your product build area, you would specify `ADDIRS=bin lib`.

`ADDFILES= "<'find' command options>"`

lists file wildcards to include or exclude with **`find(1)`** options. E.g., to exclude files ending in tilde (i.e., **`emacs`** backup files), specify `ADDFILES= ! -name '*~'.`

`ADDEMPTY="<dir1> <dir2> <dir3>..."`

lists empty directories to include in the product tar file. By default the **`tar`** command does not include empty directories in a tar file. Listing empty directories here causes them to be added.

`ADDCMD="<command>"`

specifies a command that generates a list of files on standard output. These files will then be included in the tar file. This could be used, for example, to use an explicit file inclusion list like `ADDCMD="cat my_file_list"`.

Or it could be used to specify a find command with filtering, sorting, and so on, e.g.,

```
ADDCMD= "find . ! -name '*.o' | egrep
-v \ '/foo/|/bar/' | sort -u"
```

These values are all combined by running the following sequence of commands in the Makefile:

```
    (
      for d in  .manifest.$(PROD) $(ADDEMPTY); do echo $d; done
       test -z "$(ADDDIRS)" || find $(ADDDIRS) $(PRUNECVS) !
-type d -print
    test -z "$(ADDFILES)" || find . $(PRUNECVS) $(ADDFILES) !
-type d -print
    test -z "$(ADDCMD)" || sh -c "$(ADDCMD)"
    )
```

(where PRUNECVS holds **`find`** options to prevent **`find`** from going into **CVS** directories). This generates a long list of files that get added to the tar file.

19.9 Adding your Product to a Distribution Node

The Makefile for **template_product** is set up to allow distribution to fnkits by default:

- The macro `ADDPRODUCT_HOST`, which indicates the distribution node to which products get added, is set to the default value `fnkits.fnal.gov`.
- Under the section called *Standard Product Distribution/Declaration Targets* the target `kits` is configured to add a product to *fnkits* and declare it to the `KITS` database.

To add a product to a different distribution node (e.g., *distnode.fnal.gov*):

- change the value of the macro `ADDPRODUCT_HOST` to `distnode.fnal.gov`
- add the target `distnode: addproduct` to the distribution section
- and run the **`make`** command with the new target, e.g., **`make distnode`**

19.9.1 Add Product to fnkits

Keeping the defaults in place, simply change to the directory of your product and run **`make kits`**:

```
% cd /tmp/newprod
% make kits
   rm -f /tmp/build-newprod-v1_0
   creating .manifest...
   creating /tmp/newprod/../newprodSunOS+5v1_0.tar...
   /tmp/newprod/../newprodSunOS+5v1_0.tar:
   -rw-rw-r-- mengel/oss        0 Apr  1 11:19 1998 .header
   -rw-rw-r-- mengel/oss      381 Apr  1 11:18 1998 .manifest
   -rwxrwxr-x  mengel/oss              5  Apr    1  11:07  1998
   ./ups/Version
   -rwxr-xr-x  mengel/oss             55  Apr    1  11:07  1998
   ./ups/INSTALL_NOTE
   -rwxr-xr-x  mengel/oss             43  Apr    1  11:07  1998
   ./ups/setup.csh
   -rwxr-xr-x  mengel/oss             49  Apr    1  11:07  1998
   ./ups/setup.sh
   -rwxr-xr-x  mengel/oss             43  Apr    1  11:07  1998
   ./ups/unsetup.csh
   -rwxr-xr-x  mengel/oss             49  Apr    1  11:07  1998
   ./ups/unsetup.sh
   -rwxr-xr-x  mengel/oss             15  Apr    1  11:07  1998
   ./ups/current
   -rwxr-xr-x  mengel/oss             15  Apr    1  11:07  1998
   ./ups/uncurrent
   -rwxr-xr-x  mengel/oss             15  Apr    1  11:07  1998
   ./ups/configure
   -rwxr-xr-x  mengel/oss             15  Apr    1  11:07  1998
   ./ups/unconfigure
   -rwxr-xr-x  mengel/oss            462  Apr    1  11:07  1998
   ./ups/action.table
   -rw-r--r-- mengel/oss    19858 Apr  1 11:14 1998 ./Makefile
   -rw-r--r-- mengel/oss      190 Mar 30 17:21 1998 ./README
   -rwxr-xr-x  mengel/oss             87  Feb    5  16:32  1998
   ./test/TestScript
   -rw-rw-r--  mengel/oss             36  Apr    1  11:08  1998
   ./src/hello.c
   -rw-rw-r--  mengel/oss             26  Apr    1  11:09  1998
   ./src/Makefile
   -rwxrwxr-x mengel/oss     5380 Apr  1 11:09 1998 ./src/hello
   -rwxrwxr-x mengel/oss     5380 Apr  1 11:09 1998 ./bin/hello

   upd      addproduct            -h      fnkits            -T
   "/tmp/newprod/../newprodSunOS+5v1_0.tar"  \
```

```
 -M ups  -m action.table  -U ups  -f SunOS+5
updderr::updderr_syslog - successful ups declare newprod v1_0
\
  -T  ftp://fnkits/ftp/products/newprod/v1_0/SunOS+5.tar  -f
SunOS+5 \
 -r /ftp/products/newprod/v1_0/SunOS+5 -z /ftp/upsdb -q "" \
 -M /ftp/upsdb/newprod -m v1_0.table
rm -f "/tmp/newprod/../newprodSunOS+5v1_0.tar"
```

After adding your product, use **`upd list`** to check that it arrived properly:

```
% upd list -a newprod
```

```
DATABASE=/ftp/upsdb
        Product=newprod Version=v1_0    Flavor=SunOS+5
                Qualifiers=""   Chain=""
```

19.9.2 Specify Multiple Flavors

To add different flavors of the same product without having to modify the Makefile, you may find it convenient to specify the flavor on the **`make`** command line, e.g.,

```
% make "FLAVOR=SunOS+5" kits
```

or, more generally,

```
% make "FLAVOR=${UPS_FLAVOR}" kits
```

19.10 Adding your Product Source to a CVS Repository

At this point, your product is eligible for inclusion in one of Fermilab's **CVS** repositories. This allows tracking of the software revisions, and allows other people to find it, get a particular version, and build it if they need to. The eligibility standards are described in the document *Using Fermilab CVS Product Source Repositories*, at `http://www.fnal.gov/docs/products/template_product/FermiRepository/FermiRepository.html`.

First set up **CVS** appropriately for the repository you're going to use (the example shows `fermilab`), then import your product:

```
% cvs import newprod v1_0 fermilab
```

19.11 Removing your Product from a Distribution Node

A special target is provided in the top-level Makefile to remove a product from KITS, namely:

```
unkits: delproduct
```

To remove your product from the KITS database on the *fnkits* node, just run the command:

```
% make unkits
```

```
upd delproduct -h fnkits -f SunOS+5  newprod v1_0
upderr::upderr_syslog  -  successful  ups  undeclare  newprod
v1_0 -f SunOS+5
```

If your product is on a distribution node other than *fnkits*, the Makefile has probably already been edited to recognize that node (see section 19.9 *Adding your Product to a Distribution Node*). Add a target analogous to the unkits target. For example if you have:

```
distnode:  addproduct
```

then add the target:

```
undistnode:  delproduct
```

To remove the product, run the command:

```
% make undistnode
```